Software Metrics

Kristian Sandahl
Agenda:
Terminology and scope
Quality factors revisited
Model of failure-based metrics
Usability metrics
Cyclomatic complexity
Examples of metrics and quality factors
Practical use
Requirements

System Design
(Architecture, High-level Design)

Module Design
(Program Design, Detailed Design)

Implementation
of Units (classes, procedures, functions)

Unit testing

Acceptance Test
(Release testing)

System Testing
(Integration testing of modules)

Module Testing
(Integration testing of units)

Verify Requirements, Verify Specification

Verify System Design

Verify Module Design

Verify Implementation

Project Management, Software Quality Assurance (SQA), Supporting Tools, Education

Metrics/K Sandahl

2020-09-29
Measurement - metrics

Most common use:

• Measurement – directly measured on:
  – Document, no of pages
  – Design, no of model elements
  – Code, no of lines
  – Process, iteration length
  – Quality, avg no of hours to learn a system
• Metrics – is a combination of measurements, e.g.
  number of faults found in test/hours of testing
Agenda:
Terminology and scope
Quality factors revisited
Model of failure-based metrics
Usability metrics
Cyclomatic complexity
Examples of metrics and quality factors
Practical use
Quality factors

- Correctness
- Reliability
- Efficiency
- Usability
- Integrity
- Maintainability
- Flexibility
- Testability
- Security
- Portability

- Reusability
- Interoperability
- Survivability
- Safety
- Manageability
- Supportability
- Replaceability
- Functionality

Measuring these requires both research, experience and imagination.
Agenda:
Terminology and scope
Quality factors revisited
Model of failure-based metrics
Usability metrics
Cyclomatic complexity
Examples of metrics and quality factors
Practical use
Simplified model with repair time

status

Up and running

Being repaired

0 \quad t_1 \quad t_2 \quad t_3 \quad \text{time}

TBF_1 \quad TBF_2

TTR_1 \quad TTR_2

TTF_1 \quad TTF_2 \quad TTF_3
Reliability growth model

The probability that the software executes with no failures during a specified time interval

<table>
<thead>
<tr>
<th>Failure number</th>
<th>Failure time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
</tr>
<tr>
<td>4</td>
<td>43</td>
</tr>
<tr>
<td>5</td>
<td>58</td>
</tr>
<tr>
<td>6</td>
<td>70</td>
</tr>
<tr>
<td>7</td>
<td>88</td>
</tr>
<tr>
<td>8</td>
<td>103</td>
</tr>
<tr>
<td>9</td>
<td>125</td>
</tr>
<tr>
<td>10</td>
<td>150</td>
</tr>
<tr>
<td>11</td>
<td>169</td>
</tr>
<tr>
<td>12</td>
<td>199</td>
</tr>
<tr>
<td>13</td>
<td>232</td>
</tr>
<tr>
<td>14</td>
<td>260</td>
</tr>
<tr>
<td>15</td>
<td>300</td>
</tr>
</tbody>
</table>

Approximation: MTTF/(1+MTTF)
Easier to manage: Failure intensity

Default: [failures / hours of execution time]

Or other natural unit

Another approximation: \( \lambda = \frac{(1-R)}{t} \)
Similar pattern: Availability and Maintainability

• Measure Mean Time To Repair (MTTR) and Mean Time To Failure (MTTF)

• Availability, A:
  • $A = \frac{MTTF}{MTTF + MTTR}$

• Measure Mean Time To Repair (MTTR)

• Maintainability, M:
  • $M = \frac{1}{1 + MTTR}$
Agenda:
Terminology and scope
Quality factors revisited
Model of failure-based metrics
Usability metrics
Cyclomatic complexity
Examples of metrics and quality factors
Practical use
### Measure usability?

<table>
<thead>
<tr>
<th>Relevance</th>
<th>Efficiency</th>
<th>Learnability</th>
<th>Attitude</th>
</tr>
</thead>
</table>
| - number of good and bad features recalled by users  
- number of available commands not invoked by users  
- number of available commands invoked by users  
- number of times user needs to work around a problem  
- percent of task completed | - time to complete a task  
- percent of task completed  
- percent of task completed per unit time (speed metric)  
- time spent in errors  
- number of commands used  
- frequency of help and documentation use  
- time spent using help or documentation  
- number of repetitions of failed commands | - ratio of successes to failures (over time)  
- time spent in errors  
- percent or number of errors  
- number of commands used  
- frequency of help and documentation use  
- time spent using help or documentation  
- number of repetitions of failed commands | - percent of favorable/unfavorable user comments  
- number of good and bad features recalled by users  
- number of users preferring the system  
- number of times user loses control of the system  
- number of times the user is disrupted from a work task |

---

**Metrics/K Sandahl**
Agenda:
Terminology and scope
Quality factors revisited
Model of failure-based metrics
Usability metrics
Cyclomatic complexity
Examples of metrics and quality factors
Practical use
Computation of cyclomatic complexity

Cyclomatic complexity has a foundation in graph theory and is computed in the following ways:

1. Cyclomatic complexity $V(G)$, for a flow graph, $G$, is defined as:

   $$V(G) = E - N + 2P$$

   $E$: number of edges
   $N$: number of nodes
   $P$: number of disconnected parts of the graph

2. Cyclomatic complexity $V(G)$, for a flow graph, $G$, with only binary decisions, is defined as:

   $$V(G) = b + 1$$

   $b$: number of binary decisions
Examples of Graphs and calculation of McCabe’s Complexity Metric

\[ E = 1, N = 2, P = 1 \]
\[ V = 1 - 2 + 2 = 1 \]

\[ E = 4, N = 4, P = 1 \]
\[ V = 4 - 4 + 2 = 2 \]

\[ E = 2, N = 4, P = 2 \]
\[ V = 2 - 4 + 4 = 2 \]

\[ E = 10, N = 7, P = 1 \]
\[ V = 10 - 7 + 2 = 5 \]

\[ E = 12, N = 11, P = 3 \]
\[ V = 12 - 11 + 6 = 7 \]
Control-flow

Basic block

E = 9
N = 8
P = 1

V = 3

B = number of binary decision points

V = B + 1
Agenda:
Terminology and scope
Quality factors revisited
Model of failure-based metrics
Usability metrics
Cyclomatic complexity
Examples of metrics and quality factors
Practical use
Software metrics

- Usage-based metrics
- Verification & Validation metrics
- Volume metrics
- Structural metrics
- Effort metrics
- Direct measurement
- Indirect measurement

Note: Pedagogical model only!
Usage based metrics - example

• Description: Number of good and bad features recalled by users.

• How to obtain data: Set up a test scenario. Let test users run the scenario. Collect number of good and bad features in a questionnaire afterwards.

• How to calculate the metric: Take the average of number of good and no. bad features. Two values.

• Relevant quality factor: Relevance – many good and few bad features indicates a good match with the users’ mind-set.
Verification and validation metrics - example

• Description: Rate of severe defects found in inspection of design description.

• How to obtain data: Perform an inspection according to your process. Make sure that severity is in the classification scheme.

• How to calculate the metric: Divide the number of defects classified with highest severity with total number of defects in the Inspection record.

• Relevant quality factor: Safety – a high proportion of severe defects in design indicates fundamental problems with the solution and/or competence.
Volume metrics - example

- Description: Number of non-comment lines of code.
- How to obtain data: Count non-comment lines of the code with a tool.
- How to calculate the metric: See above.
- Relevant quality factor: Reliability – it is often hard to understand a large portion of code, the fault density is often higher for large modules.
Structural metrics - example

• Description: Maximum depth of inheritance tree.
• How to obtain data: Count the depth of the inheritance tree for all classes with a tool.
• How to calculate the metric: Take the maximum value of the classes.
• Relevant quality factor: Understandability – It is hard to determine how a change in a higher class will affect inherited/overridden methods.
Effort metrics - example

• Description: Time spent in testing.
• How to obtain data: Make sure that testing activities are distinguished in time reporting forms. Make sure that all project activities are reported.
• How to calculate the metric: Sum the number of hours for all activities in testing for all people involved.
• Relevant quality factor: Testability – a comparably long testing time indicates low testability.
Agenda:
Terminology and scope
Quality factors revisited
Model of failure-based metrics
Usability metrics
Cyclomatic complexity
Examples of metrics and quality factors
Practical use
The Goal Question Metric approach

- Outside the written exam we can use a top-down approach: Goal-Question-Metric (GQM)

<table>
<thead>
<tr>
<th>Goal</th>
<th>Purpose</th>
<th>Improve the timeliness of change request processing from the project manager's viewpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue</td>
<td>Object (process)</td>
<td>Viewpoint</td>
</tr>
</tbody>
</table>

| Question | What is the current change request processing speed? |

| Metrics | Average cycle time | Standard deviation | % cases outside of the upper limit |

| Question | Is the performance of the process improving? |

| Metrics | Current average cycle time - Baseline average cycle time × 100 | Subjective rating of manager's satisfaction |

Basili, Caldiera, Rombach (1994)
## Metric Threshold Value

<table>
<thead>
<tr>
<th>Metric</th>
<th>Threshold Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Self Transitions</td>
<td>-</td>
</tr>
<tr>
<td>Transitions/State</td>
<td>Middle level state</td>
</tr>
<tr>
<td></td>
<td>4 - 5</td>
</tr>
<tr>
<td>State Depth</td>
<td>3</td>
</tr>
</tbody>
</table>

Rank = 1.2 + 0.007NonSelfTransitions + 0.17Transitions/state + 0.25StateDepth

Rezaei, Ebersjö, Sandahl, Staron  
Identifying and managing complex modules in executable software design models  
IWSM Mensura 2014 conference